

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КИЇВСЬКИЙ

ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені Ігоря Сікорського»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

**Кафедра системного програмування та спеціалізованих комп’ютерних систем**

**Лабораторна робота №2**

з дисципліни **Бази даних і засоби управління**

*на тему: “Створення додатку бази даних, орієнтованого на взаємодію з СУБД PostgreSQL”*

Виконав:

студент 3 курсу

групи КВ-91

Зиков А.С.

Перевірив:

Павловський В. І.

Київ – 2021

# Постановка задачі

*Метою роботи* є здобуття вмінь програмування прикладних додатків баз даних PostgreSQL.

*Загальне завдання* роботи полягає у наступному:

1. Реалізувати функції перегляду, внесення, редагування та вилучення даних у таблицях бази даних, створених у лабораторній роботі №1, засобами консольного інтерфейсу.
2. Передбачити автоматичне пакетне генерування «рандомізованих» даних у базі.
3. Забезпечити реалізацію пошуку за декількома атрибутами з двох та більше сутностей одночасно: для числових атрибутів – у рамках діапазону, для рядкових, – як шаблон функції LIKE оператора SELECT SQL, для логічного типу – значення True/False, для дат – у рамках діапазону дат.
4. Програмний код виконати згідно шаблону MVC (модель-подання- контролер).

**Інформація про програму**

Посилання на репозиторій у GitHub з вихідним кодом програми та звітом: https://github.com/Luritron/Data-Base-KV-91-Zykov-Andrii

Використана мова програмування: Python 3.10 (IDE PyCharm)

Використані бібліотеки: psycopg2 (для зв’язку з СУБД), time (для виміру часу запиту пошуку для завдання 3), та шаблон проектування MVC.

**Модель «сутність-зв’язок» для галузі сервіс “питання-відповіді”**

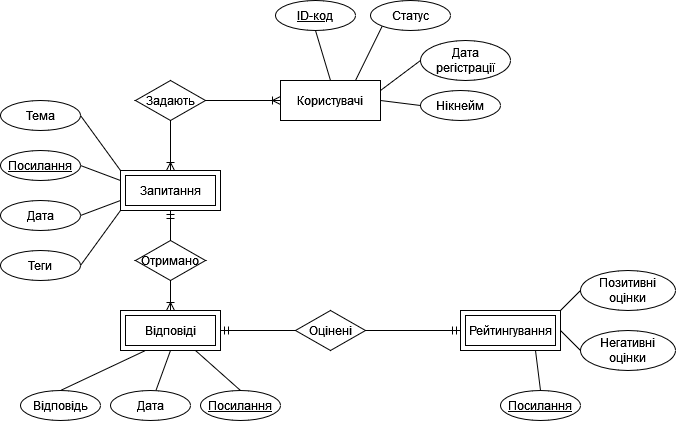


Рисунок 1. ER-діаграма побудована за нотацією Чена

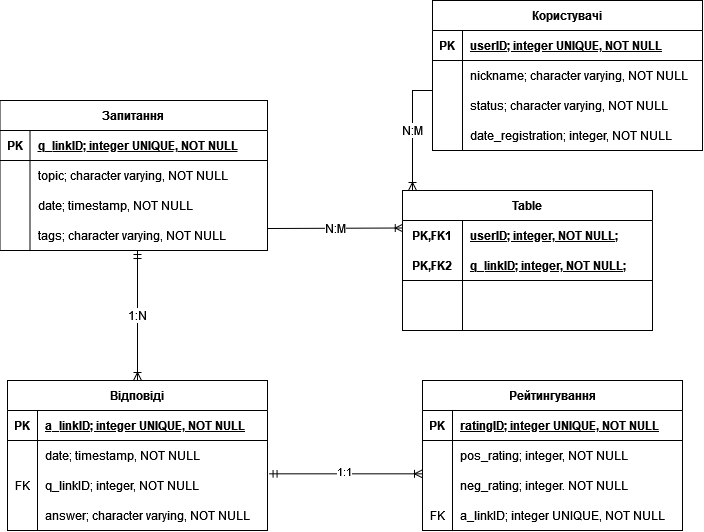


Рисунок 2. Схема бази даних

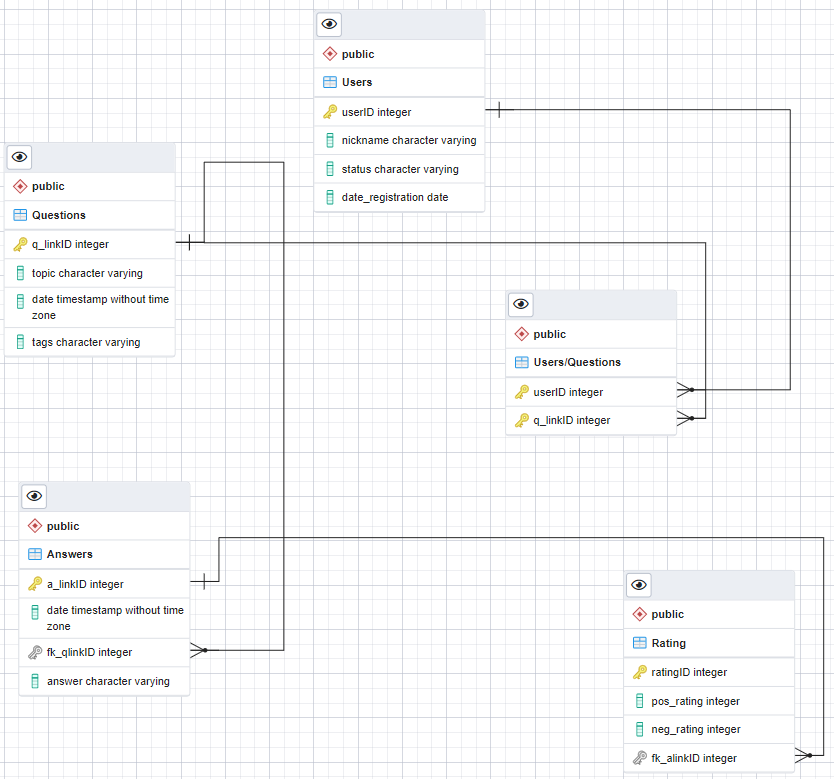
****

Рисунок 3. Схема бази даних у pgAdmin4

Таблиця1. Опис структури бази даних

|  |  |  |
| --- | --- | --- |
| Сутність | Атрибут | Тип атрибуту |
| **Users -** *містить у собі основні дані про користувача* | **userID -** *унікальний ідентифікатор користувача*  **nickname -** *містить дані про найменування користувача*  **status -** *містить дані, чи перебуває зараз користувач на сайті*  **date\_registration -** *містить дані про дату народження користувача* | **integer -** *числовий*  **character varying -** *рядок*  **character varying -** *рядок*  **integer -** *числовий* |
| **Users/Questions -** *містить у собі проміжні дані про ключі кожної з відповідних таблиць (використовується для реалізації N:M)* | **userID -** *ідентифікатор користувача, що відсилає нас на відповідну таблицю Users*  **q\_linkID -** *ідентифікатор запитання, що відсилає нас на відповідну таблицю Questions* | **integer -** *числовий*  **integer -** *числовий* |
| **Questions -** *містить у собі основні дані про створене запитання* | **q\_linkID -** *унікальний ідентифікатор запитання*  **topic -** *містить дані про тему запитання*  **date -** *містить дані про час коли було створено запитання*  **tags -** *містить короткі дані за якими простіше знаходити спорідненні запитання* | **integer -** *числовий*  **character varying -** *рядок*  **timestamp without time zone** - *мітка часу*  **character varying -** *рядок* |
| **Answers -** *містить у собі основні дані про відповідь дану на запитання* | **a\_linkID -** *унікальний ідентифікатор відповіді*  **date -** *містить дані про час коли було створено відповідь*  **fk\_qlinkID -** *ідентифікатор запитання, на яке дано відповідь*  **answer -** *містить дані про надану на запитання відповідь* | **integer -** *числовий*  **timestamp without time zone** - *мітка часу*  **integer -** *числовий*  **character varying -** *рядок* |
| **Rating -** *містить у собі дані про оцінене запитання* | **ratingID -** *унікальний ідентифікатор оцінки*  **pos\_rating -** *позитивна оцінка*  **neg\_rating -** *негативна оцінка*  **fk\_alinkID -** *ідентифікатор відповіді, яка була оцінена* | **integer -** *числовий*  **integer -** *числовий*  **integer -** *числовий*  **integer -** *числовий* |

Схема відповідає 1НФ, тому що:

1) Кожна таблиця має основний ключ (Primary key, PK) та мінімальний набір атрибутів, які ідентифікують запис;

2) Кожен атрибут повинен має лише одне значення, а не множину значень.

Схема відповідає 2НФ, тому що:

1) Вона відповідає вимогам 1НФ;

2) Кожен не ключовий атрибут функціонально залежить від основного атрибуту.

Схема відповідає 3НФ, а саме:

1) Вона відповідає вимогам 2НФ;

2) Дані в таблиці залежать винятково від основного ключа

**Структура програми**

За шаблоном проектування MVC, додаток складається з таких модулей:

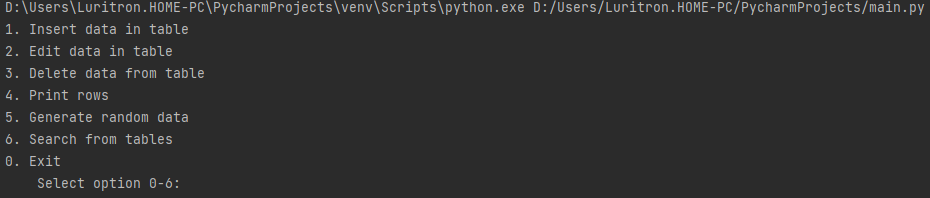
*model.py* — виконує операції з БД – SELECT, INSERT, DELETE, UPDATE та, більш складні операції з БД

*controller.py* — головне та допоміжні меню, для зручного керуванням БД

*config.py* — файл з вхідними даними до БД

*main.py* — точка входу в програму, та підключення до БД

# Схема меню користувача

На скріншоті відображено невеличне меню, що реалізує можливість обрати певну операцію для оперування над базою даних. Кожна команда запускає відповідну функцію з файлу Controller, яка в подальшому передає аргументи у функцію файлу Model, яка в свою чергу сформовує і здійснює запит до бази даних

Реалізовані такі методи:

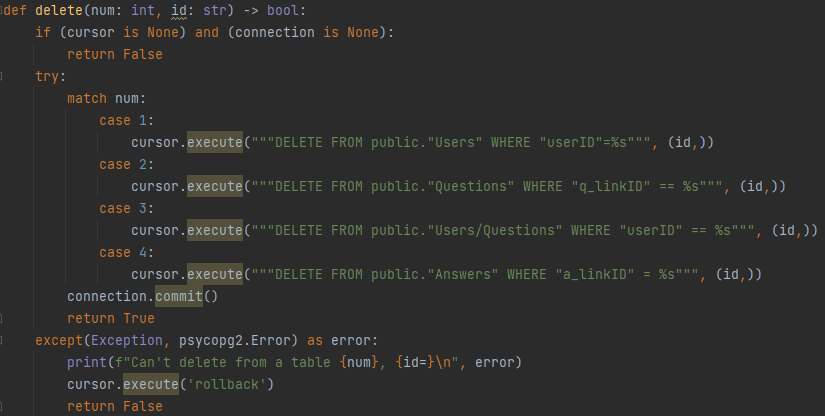
1. Insert data in table — викликає функцію вставки даних у таблицю;
2. Edit data in table — викликає функцію редагування даних у таблиці;
3. Delete data from table — викликає функцію видалення даних у таблиці;
4. Print rows — викликає функцію виводу даних з таблиці.

Кожна з функцій викликає допоміжну функцію select\_table() яка дозволяє користувачеві вибрати таблицю

1. Generate random data — дозволяє користувачеві заповнити таблицю або всю базу даних випадково згенереваними данними.
2. Search from tables — викликає функцію пошуку даних у таблицях за атрибутами та поєднання таблиць за ключем.

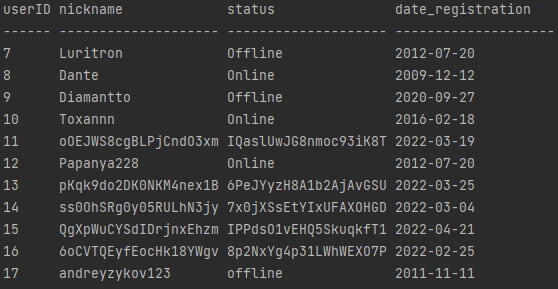
# Завдання 1

***Запит на видалення***

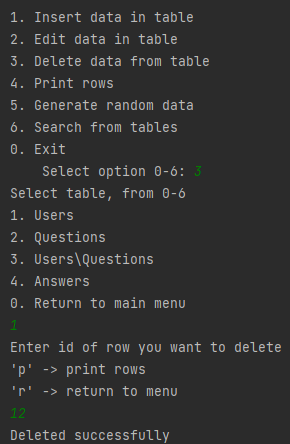
******

Реалізуємо видалення даних з таблиці Users.

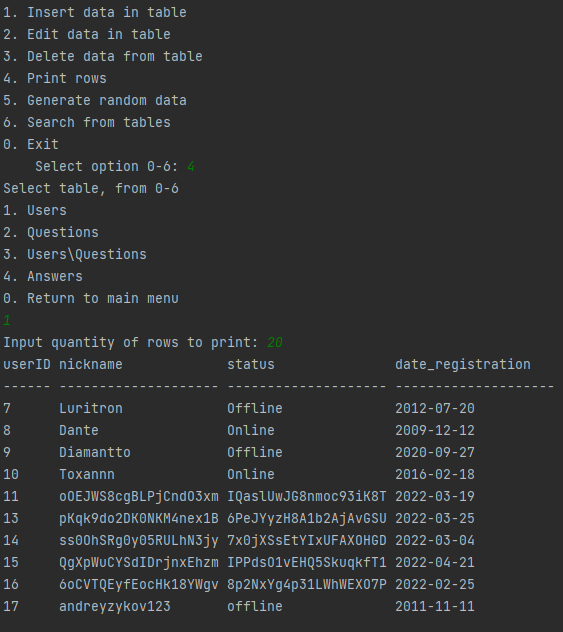
Таблиця Users до видалення:



У даному випадку видалятиметься лише рядок із даними, що повязаний до визначенного id. Для демонсрації роботи, видалимо 12 користувача за id-кодом. Таким чином:



Після видалення таблиця набула вигляду:

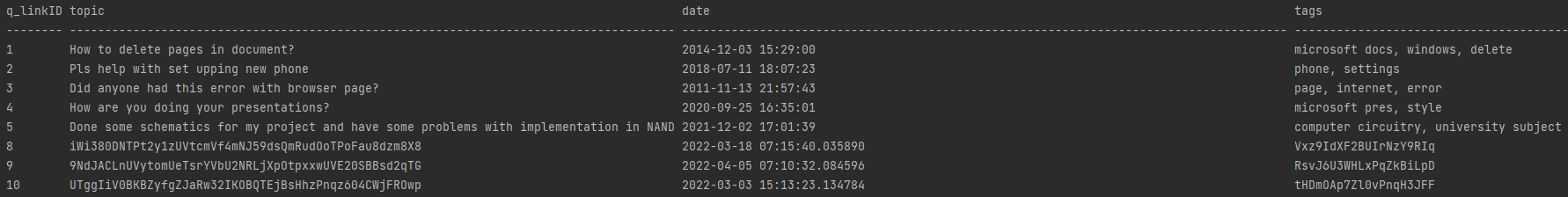


## Запит на вставку поля

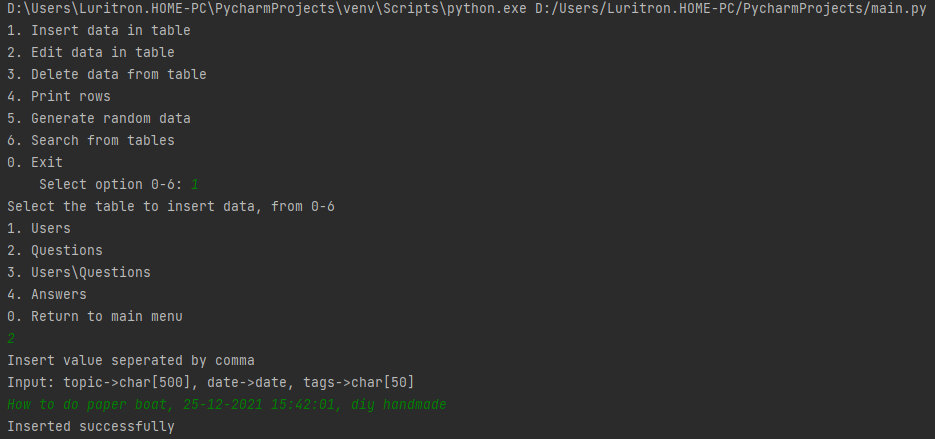
## pycharm64_A6DVO3Gw2I

Для перевірки роботи розглянемо запити на вставки в таблицю Questions.

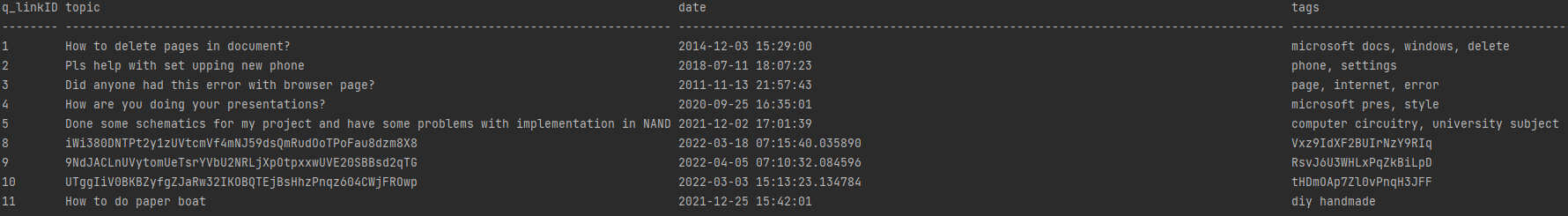
Таблиця Questions до вставки:



Виконаємо вставку:



Таблиця Questions після вставки:

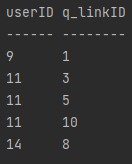


***Запит на зміну полів***

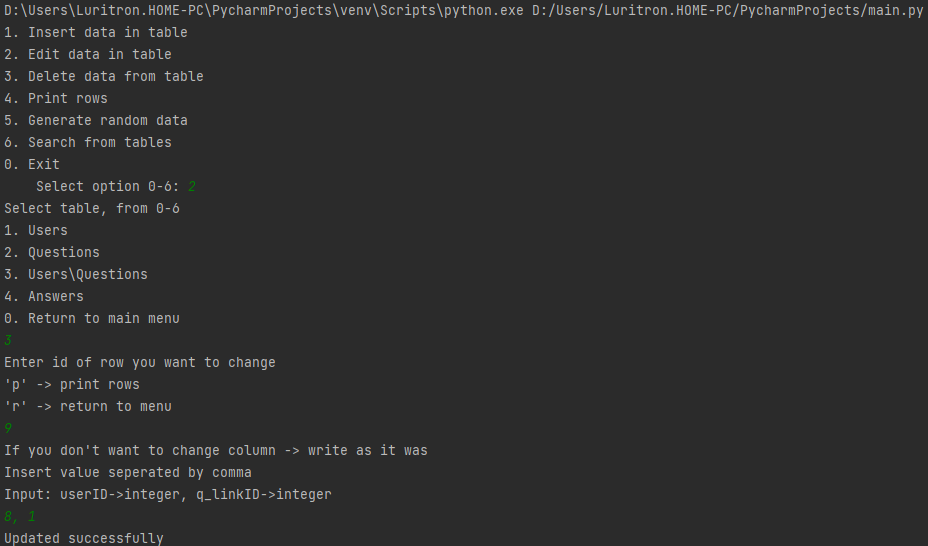
## pycharm64_W1p4aULjBQ

Для перевірки роботи розглянемо запити на зміну значенння в таблиці Users/Questions. Спочатку коректний, потім з неіснуючим значенням зовнішнього ключа таблиці Users.

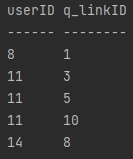
Таблиця Users/Questions до зміни:



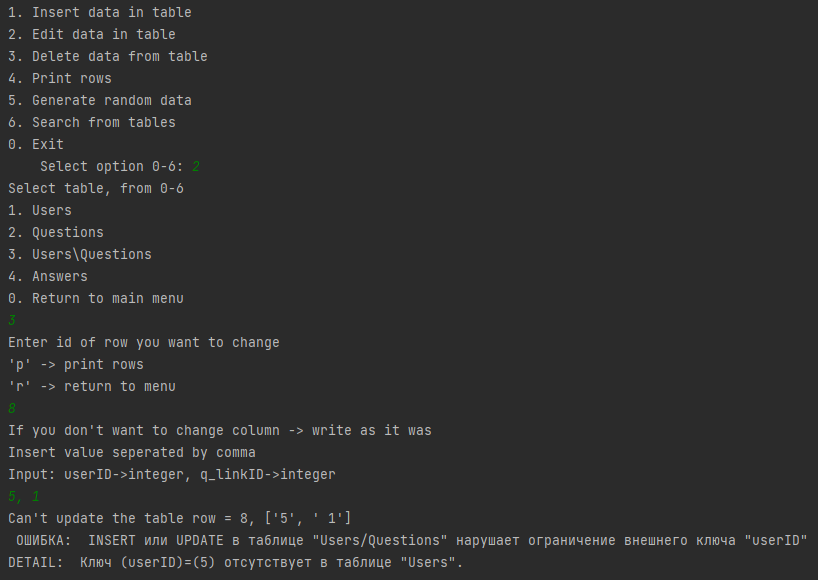
Змінимо в Users/Questions, userID з 9 на 8, а код запитання залишимо попереднім:



Таблиця Users/Questions після зміни даних:



Тепер спробуємо замінити userID = 8 у нашій таблиці на не існуюче значення userID = 5:



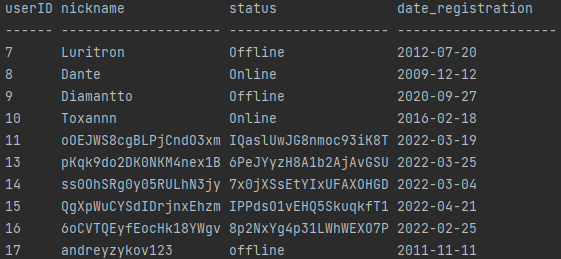
Можемо бачити помилку, про відсутність ключа 5.

# Завдання 2

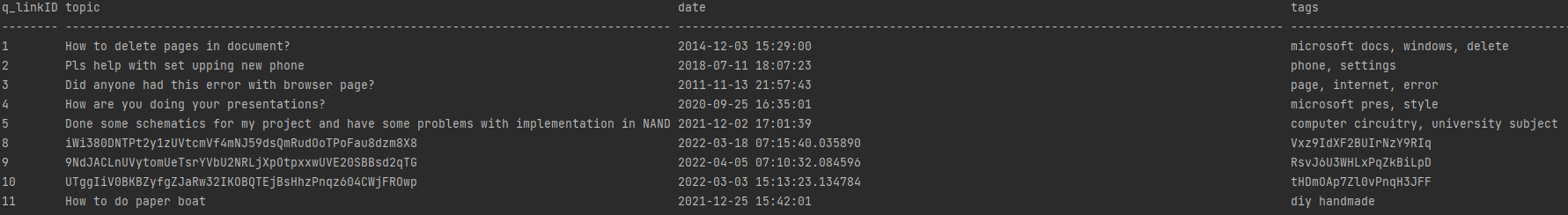
# pycharm64_eIK23IA45i

Вставка 2 псевдорандомізованих записів у кожну з таблиць.

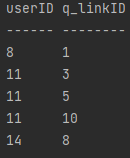
Таблиця Users перед заповненням:



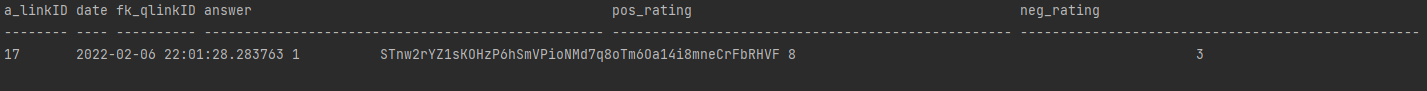
Таблиця Questions перед заповненням:



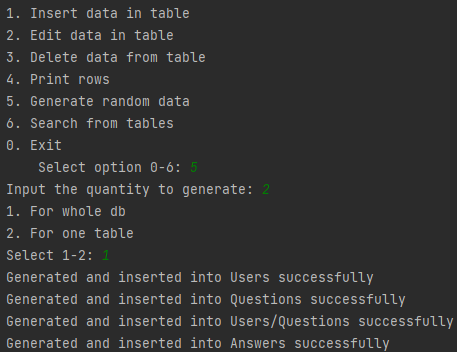
Таблиця Users/Questions перед заповненням:



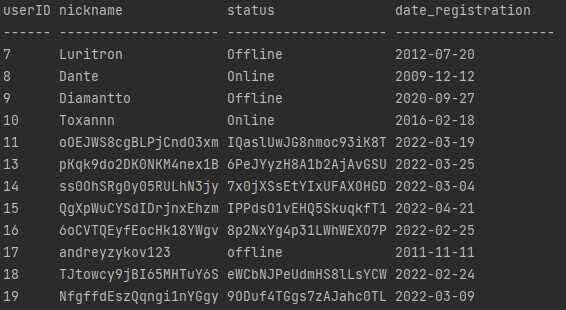
Таблиця Answers перед заповненням:



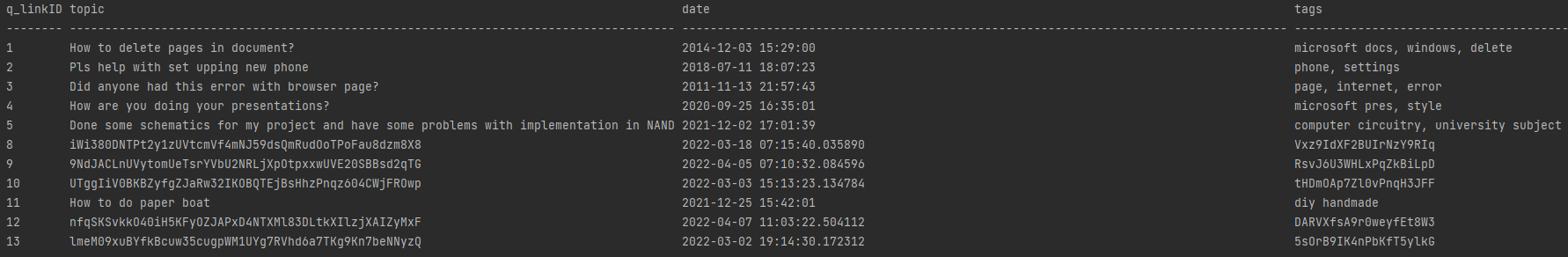
Таблиці після вставки згенерованих даних



Таблиця Users після заповнення:



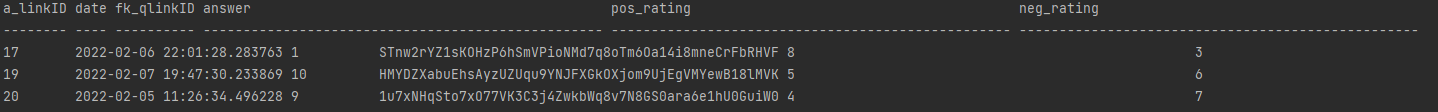
Таблиця Questions після заповнення:



# Таблиця Users/Questions після заповнення:

# pycharm64_POeEaKvgbc

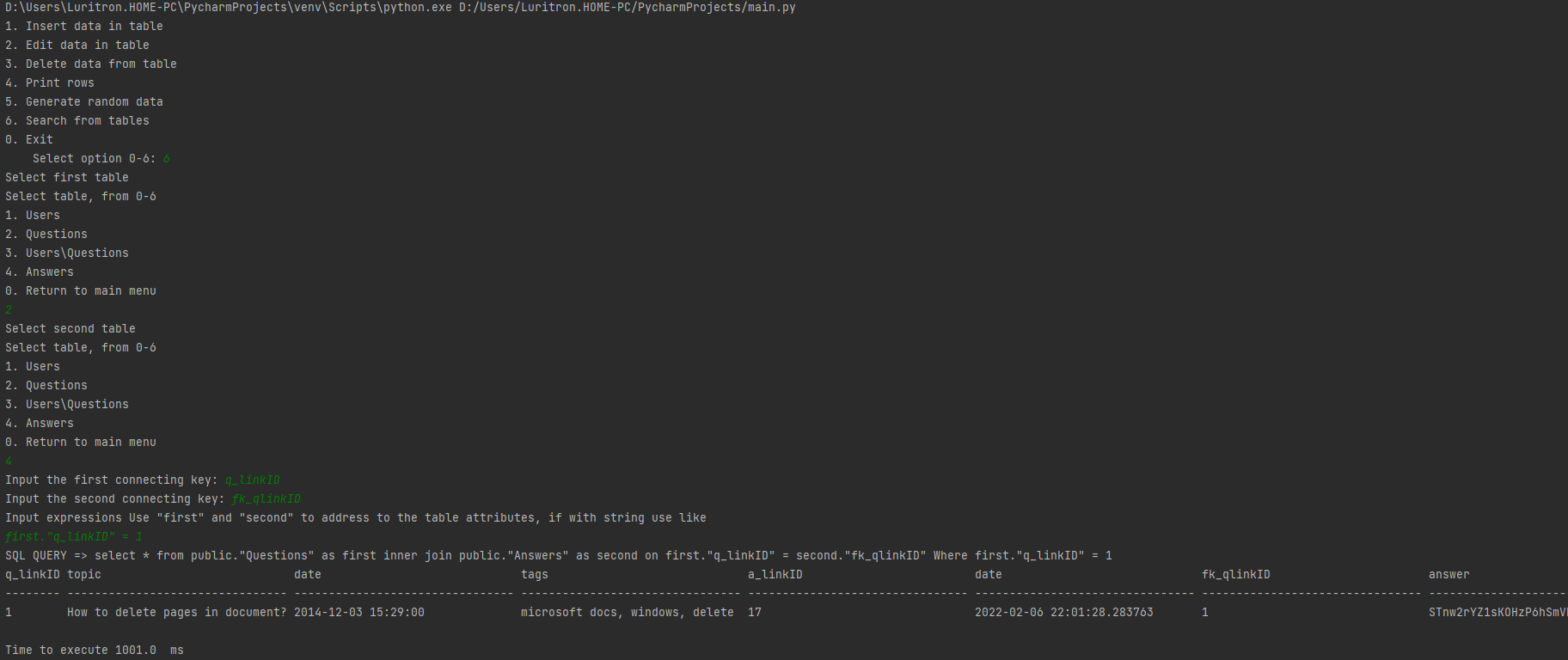
# Таблиця Answers після заповнення:



# Завдання 3

# pycharm64_knRmRAq8TU

Об’єднання двох таблиць і пошук в них за 1 атрибутом для прикладу:

****

# Завдання 4

Код програмного модулю “model.py”

import time

from config import \*

import psycopg2

cursor = None

connection = None

def connect():

try:

global connection, cursor

connection = psycopg2.connect(user=user, password=password, host='localhost', port="5432", database=database)

cursor = connection.cursor()

except (Exception, psycopg2.Error) as error:

print("Can't connect to DB", error)

disconnect()

def disconnect():

if connection:

cursor.close()

connection.close()

print("Successfully disconnected from DB")

else:

print("Can't disconnect")

def insert(num: int, col: list) -> bool:

if (cursor is None) and (connection is None):

return False

try:

match num:

case 1:

cursor.execute("""INSERT INTO PUBLIC."Users" (nickname, status, date\_registration) \

VALUES (%s, %s, %s)""", col)

case 2:

cursor.execute("""INSERT INTO PUBLIC."Questions" (topic, date, tags) \

VALUES (%s, %s, %s)""", col)

case 3:

cursor.execute("""INSERT INTO PUBLIC."Users/Questions" (userID, q\_linkID) \

VALUES (%s, %s)""", col)

case 4:

cursor.execute("""INSERT INTO PUBLIC."Answers" (date, fk\_qlinkID, answer, pos\_rating, neg\_rating) \

VALUES (%s, %s, %s)""", col)

connection.commit()

except (Exception, psycopg2.Error) as error:

print("Can't insert into table", error)

cursor.execute('rollback')

return False

return True

def select(num: int, quantity: int = 100, offset: int = 0, id: str = "") -> list:

if (cursor is None) and (connection is None):

return []

try:

match num:

case 1:

if id:

cursor.execute("""SELECT \* FROM public."Users" WHERE "userID"=%s""", id)

else:

cursor.execute("""SELECT \* FROM public."Users" ORDER BY "userID" ASC limit %s offset %s""",

(quantity, offset,))

case 2:

if id:

cursor.execute("""SELECT \* FROM public."Questions" WHERE "q\_linkID"=%s""", id)

else:

cursor.execute(

"""SELECT \* FROM public."Questions" ORDER BY "q\_linkID" ASC limit %s offset %s""",

(quantity, offset,))

case 3:

if id:

cursor.execute("""SELECT \* FROM public."Users/Questions" WHERE "userID"=%s""", id)

else:

cursor.execute("""SELECT \* FROM public."Users/Questions" ORDER BY "userID" ASC limit %s offset %s""",

(quantity, offset,))

case 4:

if id:

cursor.execute("""SELECT \* FROM public."Answers" WHERE "a\_linkID"=%s""", id)

else:

cursor.execute(

"""SELECT \* FROM public."Answers" ORDER BY "a\_linkID" ASC limit %s offset %s""",

(quantity, offset,))

return cursor.fetchall()

except(Exception, psycopg2.Error) as error:

print(f"Can't select from a table {num}, with {id=}\n", error)

return []

def delete(num: int, id: str) -> bool:

if (cursor is None) and (connection is None):

return False

try:

match num:

case 1:

cursor.execute("""DELETE FROM public."Users" WHERE "userID"=%s""", (id,))

case 2:

cursor.execute("""DELETE FROM public."Questions" WHERE "q\_linkID" = %s""", (id,))

case 3:

cursor.execute("""DELETE FROM public."Users/Questions" WHERE "userID" = %s""", (id,))

case 4:

cursor.execute("""DELETE FROM public."Answers" WHERE "a\_linkID" = %s""", (id,))

connection.commit()

return True

except(Exception, psycopg2.Error) as error:

print(f"Can't delete from a table {num}, {id=}\n", error)

cursor.execute('rollback')

return False

def update(num: int, col: list, id: int) -> bool:

if (cursor is None) and (connection is None):

return False

try:

match num:

case 1:

cursor.execute(

"""UPDATE PUBLIC."Users" SET nickname = %s, status = %s, date\_registration = %s WHERE "userID"=%s;""",

(\*col, id,))

case 2:

cursor.execute(

"""UPDATE PUBLIC."Questions" SET topic = %s, date = %s, tags = %s WHERE "q\_linkID" = %s;""",

(\*col, id,))

case 3:

cursor.execute(

"""UPDATE PUBLIC."Users/Questions" SET "userID" = %s, "q\_linkID" = %s WHERE "userID" = %s;""",

(\*col, id,))

case 4:

cursor.execute(

"""UPDATE PUBLIC."Answers" SET date = %s, "fk\_qlinkID" = %s, answer = %s, pos\_rating = %s, neg\_rating = %s, WHERE "a\_linkID" = %s;""",

(\*col, id,))

connection.commit()

except (Exception, psycopg2.Error) as error:

print(f"Can't update the table row = {id}, {col}\n", error)

cursor.execute('rollback')

return False

return True

def generate(num: int, quant: int):

if (cursor is None) and (connection is None):

return False

try:

for i in range(quant):

match num:

case 1:

cursor.execute(

"""INSERT INTO public."Users"(nickname, status, date\_registration) Select random\_string(20), random\_string(20), NOW() + (random() \* (NOW()+'90 days' - NOW())) + '30 days';""")

case 2:

cursor.execute(

"""INSERT INTO public."Questions"(topic, date, tags) Select random\_string(50), NOW() + (random() \* (NOW()+'90 days' - NOW())) + '30 days', random\_string(20); """)

case 3:

cursor.execute(

"""INSERT INTO public."Users/Questions"("userID", "q\_linkID") Select "userID", "q\_linkID" From public."Users" cross join public."Questions" order by random() limit 1;""")

case 4:

cursor.execute(

"""INSERT INTO public."Answers"(answer, pos\_rating, neg\_rating, date, "fk\_qlinkID") Select random\_string(50), random\_between(1, 10),random\_between(1, 10), ( NOW() + (random() \* (NOW()+'90 days' - NOW())) + '30 days'), "q\_linkID" From public."Questions" order by random() limit 1;""")

connection.commit()

except (Exception, psycopg2.Error) as error:

print(f"Can't insert in the table row = {num}\n", error)

cursor.execute('rollback')

return False

return True

def search(tables: list[int], key1: str, key2: str, expression: str):

sql\_query = "select \* from "

match tables[0]:

case 1:

sql\_query += 'public."Users" as first'

case 2:

sql\_query += 'public."Questions" as first'

case 3:

sql\_query += 'public."Users/Questions" as first'

case 4:

sql\_query += 'public."Answers" as first'

sql\_query += ' inner join '

match tables[1]:

case 1:

sql\_query += 'public."Users" as second'

case 2:

sql\_query += 'public."Questions" as second'

case 3:

sql\_query += 'public."Users/Questions" as second'

case 4:

sql\_query += 'public."Answers" as second'

sql\_query += f' on first."{key1}" = second."{key2}" Where {expression}'

print('SQL QUERY =>', sql\_query)

global cursor

try:

timer = time.time\_ns()

cursor.execute(sql\_query)

except (Exception, psycopg2.Error) as error:

print("Can't execute search\n", error)

return []

rows = cursor.fetchall()

timer = time.time\_ns() - timer

return rows, timer

Даний модуль є точкою доступу до бази даних з програми. Саме в ньому реалізуються всі запити. Для цього в ньому використовується бібліотека – psycopg2.

Функція connect() – пробує підключитись до БД.

Функція disconnect() – пробує відключитись від БД.

Функція insert(num, col) – пробує вставити в таблицю num, дані col.

ункція delete(num, id) – пробує видали з таблиці num, рядок з первинним ключем == id.

Функція update(num, col, id) – пробує змінити в таблиці num, рядок з первинним ключем == id, данними col.

Функцію select(num, quantity, offset, id) – пробує взяти дані з таблиці num та відсортувати їх, а якщо вказано id, то взяти тільки рядок з первинним ключем == id.

Функція generate(num, quant) – генерує дані і пробує вставити в таблицю num, quant разів ці даніазрр.

Функція search(tables, key1, key2, expression) ­– пробує об’єднати і взяти дані з таблиць